Tuning Software with Apple's Performance Tools

Session 305

Tuning Software with Apple's Performance Tools

Dave Payne
Performance Tools Manager

Session Goals

- Performance analysis concepts
- Use of Apple's performance analysis tools
  - High-level tools
  - CHUD processor-level tools
  - Integration with Xcode
- Refer to Performance documentation!

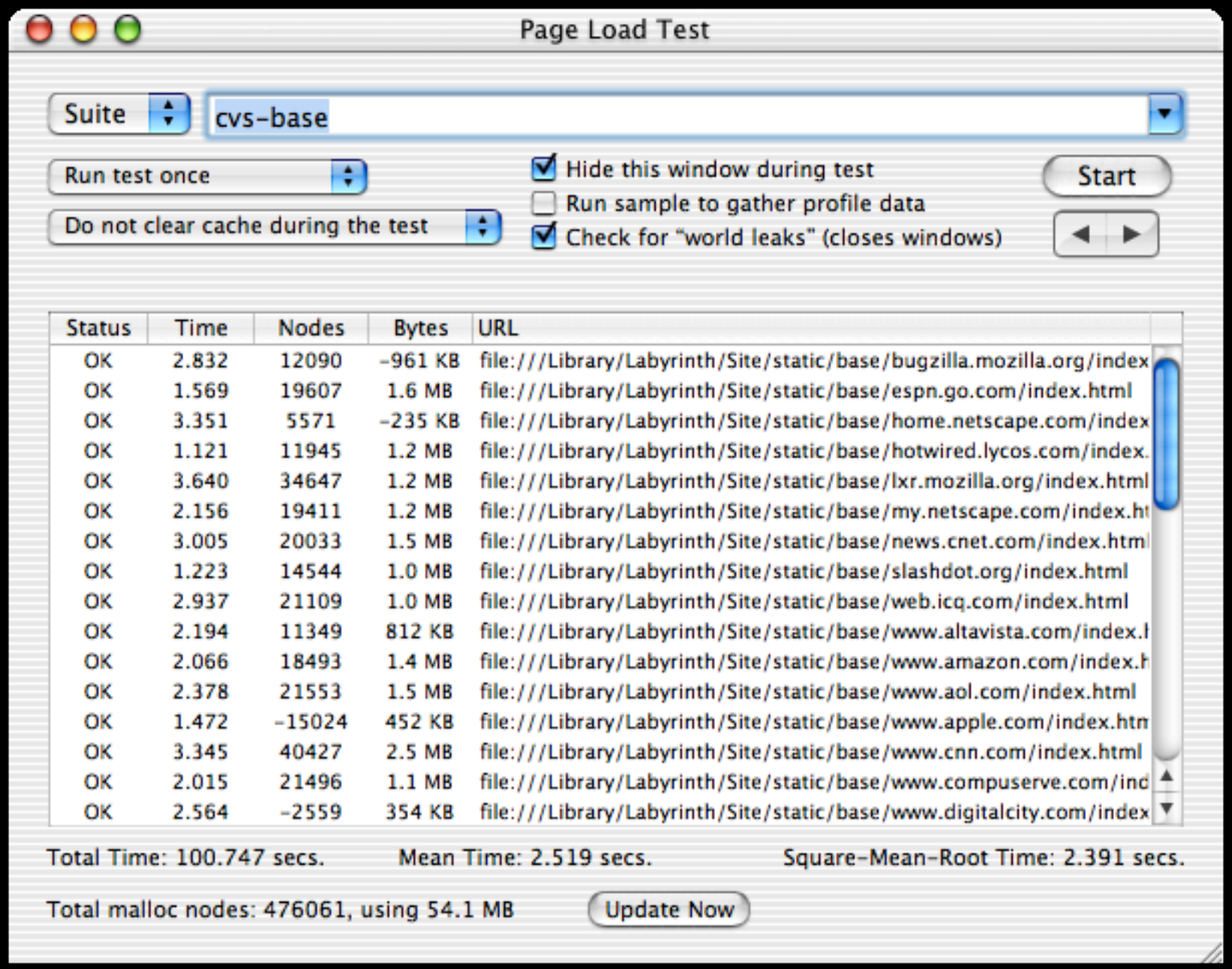
Why Worry About Performance?

- Performance is a selling point
- Performance problems may go unnoticed
- You’re not the only application on the system
- It can be hard to graft performance on at the end

A Performance Engineering Process

1. Identify common use cases
2. Define performance goals
   - Responsiveness, throughput, scalability
3. Establish benchmarks
4. Add instrumentation
5. Track results throughout your development
   - Don’t allow performance regressions
6. Focus tuning efforts on hot spots

Safari's Embedded Instrumentation



Some Areas for Benchmarking

- Private memory:
  - Static memory use and leaks
  - Dynamic memory use
- CPU:
  - Launch time
  - CPU use when idle and during major operations
  - Any UI “spins” during normal use?
- Drawing:
  - Excess redraws
  - Smooth resizing

Tools for Examining Performance

	Memory Use	Execution Behavior	Resource Use
Monitor	Activity Monitor heap top	Activity Monitor Thread Viewer Spin Control top	fs_usage sc_usage Quartz Debug
Analyze	MallocDebug ObjectAlloc vmmap	Sampler Shark MONster	Sampler OpenGL Profiler

Levels of Design Abstraction

Application Architecture	Sampler Thread Viewer MallocDebug ObjectAlloc
Module Design	
Interaction with OS	CHUD (Shark)
Bottleneck Routines	
Hardware	Activity Monitor Shark
Interaction with Other Processes	

New Tools and Features

- Activity Monitor
- Spin Control
- Integration of performance tools with Xcode
- New features in Sampler
  - Event trace graph and table
  - Outline view
  - Saving and opening traces and samples
- CHUD enhancements
  - New features
  - Tuning for G5

Software Tuning on Mac OS X:
A Case Study

Robert Bowdidge
Performance Tools Engineer

Example Program: Sketch

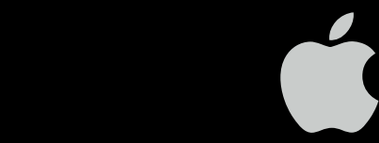
- Works great for small line drawings
- What if we wanted to use it for architectural drawings?
  - Thousands of objects, not tens
  - What happens to memory use?
  - What happens to speed?

Starting Point: Use Activity Monitor

- Provides quick numbers describing system state
  - CPU, memory usage of tasks, paging rate
- Watch your process
  - Most useful metrics: CPU usage, Private Memory
- Watch other relevant processes
  - e.g., Window Server

What if Memory Use Seems High?

- Touching too much memory is bad
  - Application is slower because of cache issues
  - Unused data just fills memory
  - If memory is scarce, this becomes disk accesses
- Worry about used memory, leaked memory
-
- ObjectAlloc: What objects are filling memory?
- MallocDebug: Which functions allocate most memory?



ObjectAlloc: Analyzing Object Use

- Shows how many objects exist
  - Current, max, total created over life of program
  - Shows totals, bar graph, backtraces
- Great for summary, noting trends
  - Did you expect 100 FooBars?
- Works for CoreFoundation, ObjC objects
  - Backtraces for Objective-C retains and releases
  - Also shows malloc blocks

What if CPU Use Seems High?

- Costly routines hurt you and the rest of the system
  - Your operations take too long
  - Other apps don’t get their share of CPU
- Sampler: What code is commonly running?
- Dynamic memory footprint: Why do I call malloc?
- Spin Control: Why did the UI just freeze there?
- QuartzDebug: When am I drawing?

Sampler: Where Are the Hot Spots?

- Shows what code is executing
  - Stops program at interval, records call stack
  - Shows only functions seen at sampling time
  - Presents call graph with sample counts
- If you need better detail, check out Shark or gprof

Know What to Measure? Instrument!

- For specific measurements, add instrumentation
  - Easy to gather numbers on each run
  - Easy to watch for regressions
- Timing APIs (choose depending on needed accuracy)
  - Nanosecond resolution:
    - `UpTime()` (CoreServices: Uses CPU time registers)
  - Microsecond resolution (plus overhead)
    - `gettimeofday()` (BSD)
    - `-[NSDate timeIntervalSinceDate:]` (Foundation)

Example: How Long Does Copy Take?

# of Items	Order List	Sketch Clipping	TIFF Clipping	PDF Clipping
500	0.02 sec	0.60 sec	0.22 sec	0.65 sec
1,000	1.74 sec	1.03 sec	0.40 sec	0.93 sec
2,000	10.28 sec	2.39 sec	0.81 sec	1.76 sec
4,000	807.06 sec	6.93 sec	2.66 sec	5.43 sec

Finding Patterns and Repetition

- Large programs involve multiple layers, hidden details
- Innocuous calls can have unexpected costs
- Do layers mistakenly repeat expensive operations?
- Example: How much memory is allocated and freed?
- New Sampler view: “trace view”
  - Shows sequence of calls to malloc (or other functions)
  - Graph of call stack depth for each sample
  - Watch for patterns of allocations and deletions

Spin Control: Catch Misbehavior

- Spinning cursor appears if app is unresponsive
- Most likely a real problem—app doing too much work
- Difficult to catch yourself
-
- Spin Control automatically samples the program
  - One sample report per unresponsive episode
  - Leave it running to capture spins
  - Sample reports can be browsed, copied, mailed

Other Tools to Check Out Yourself

- Heap memory use
  - MallocDebug: Browse memory use by allocation site
  - heap, leaks, malloc_history
- Process state
  - Thread Viewer: timeline of program state
  - sample: command-line version of Sampler
  - time: overall time for specific application
- Resources
  - fs_usage: Which file system calls are made?
  - sc_usage: Which system calls are used?

CHUD

Computer Hardware Understanding Developer Tools

Nathan Slingerland
Architecture Performance Group

Introduction to CHUD Tools

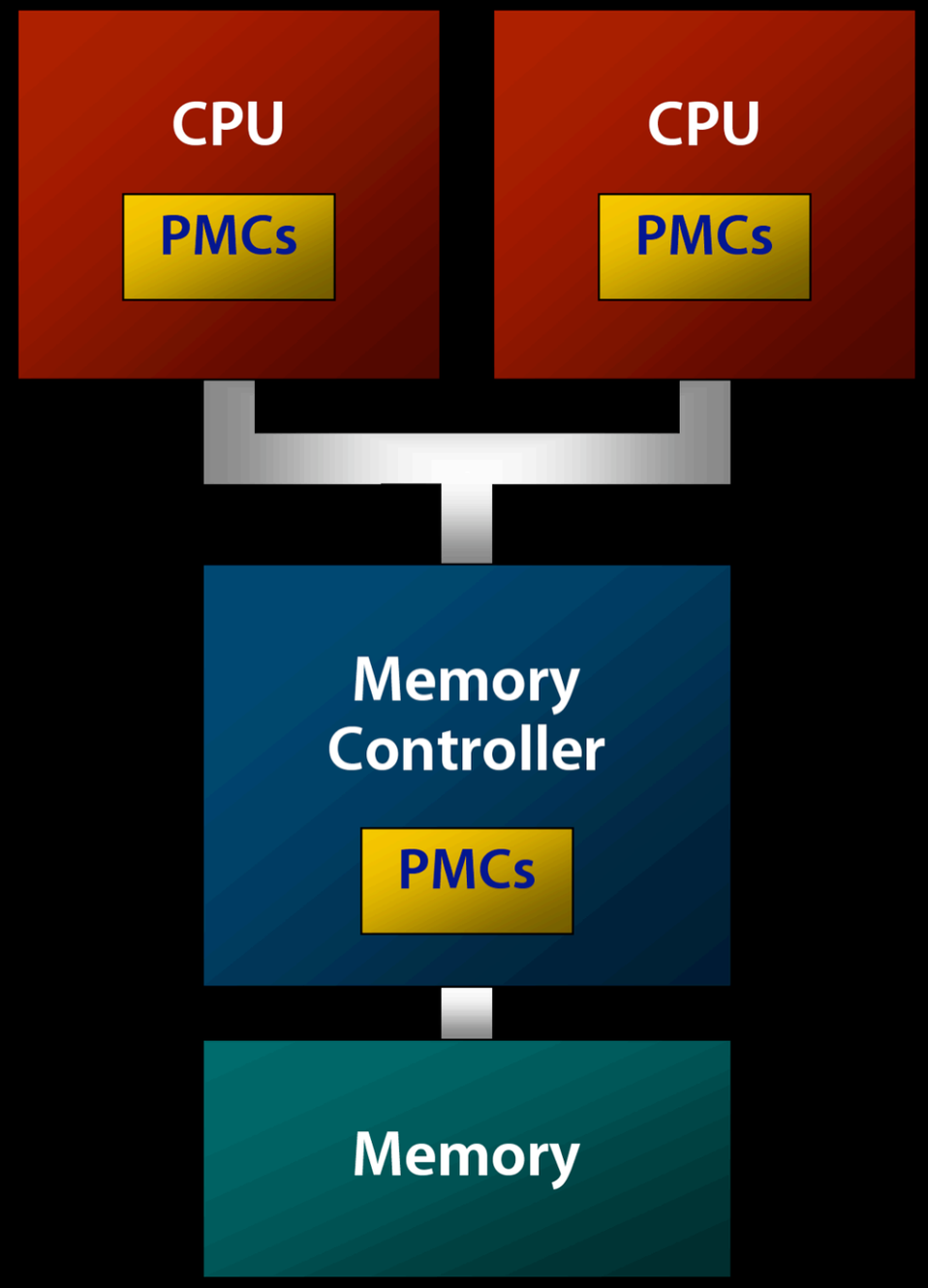
- Suite of performance analysis tools
- Low-level access to performance monitors
  - Processor
  - Memory controller
  - Operating system
- Helps you find problems and improve your code
- Freely available on the web and the Developer Tools CD!

CHUD Tools 3.0

- Profiling
  - Shark: Instruction-level profiler
  - MONster: Spreadsheet for performance events
  - Saturn: Function call graph visualizer
- Tracing
  - Amber: Instruction tracer
  - Acid: Trace analyzer
  - SimG4 and SimG5: Cycle-accurate CPU simulators
- Instrumentation
  - CHUD.framework: API for performance tuning

Performance Counters

- Apple systems have performance monitor counters (PMCs)
- PMCs record performance “events”
  - Cache misses
  - Instructions completed
  - Execution stall cycles
  - Page faults
- CHUD Tools configure, control and view the PMCs

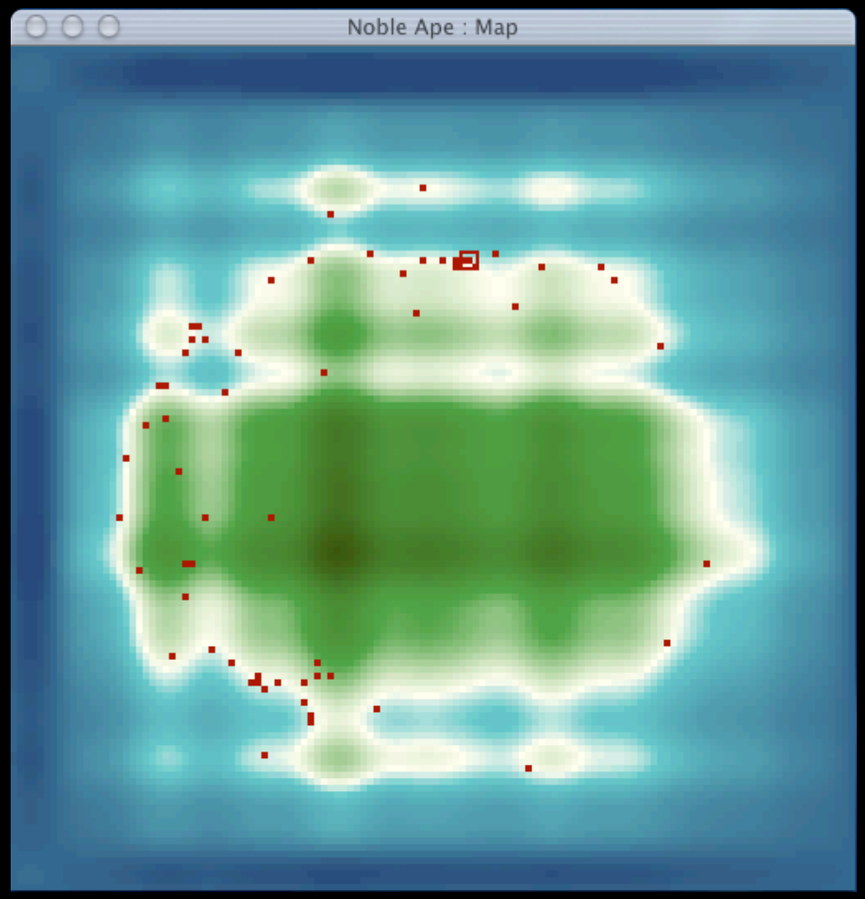


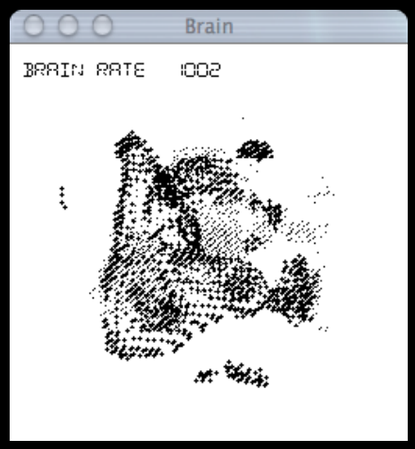
Shark

- Profile a process, thread or entire system
  - Time profile: Visualize performance hot spots
  - Event profile: Relate hardware events to source
  - Capture everything: Drivers, kernel, applications
  - Very low overhead
- Automated analysis
  - Annotated source code and disassembly
  - Static analysis to find sub-optimal code
  - Optimization tips
- Command line version
- Save and review sessions

Noble Ape Simulation

- Tom Barbalet's simulation of apes on a tropical island
  - Biological environment
  - Cognitive processes
- Open source
- For more information:
  - http://www.nobleape.com/sim/





Demo

Noble Ape Performance Summary

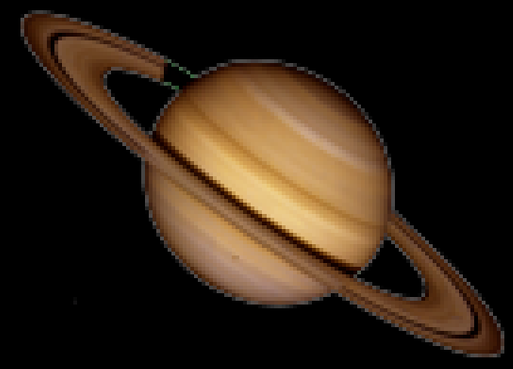
	G4 Dual 1.42GHz	G5 Dual 2GHz
Scalar	1200	1330
Threaded	2080	2490
Vector	4510	10200
Vector Optimized	5700	16140

Ape thoughts per second

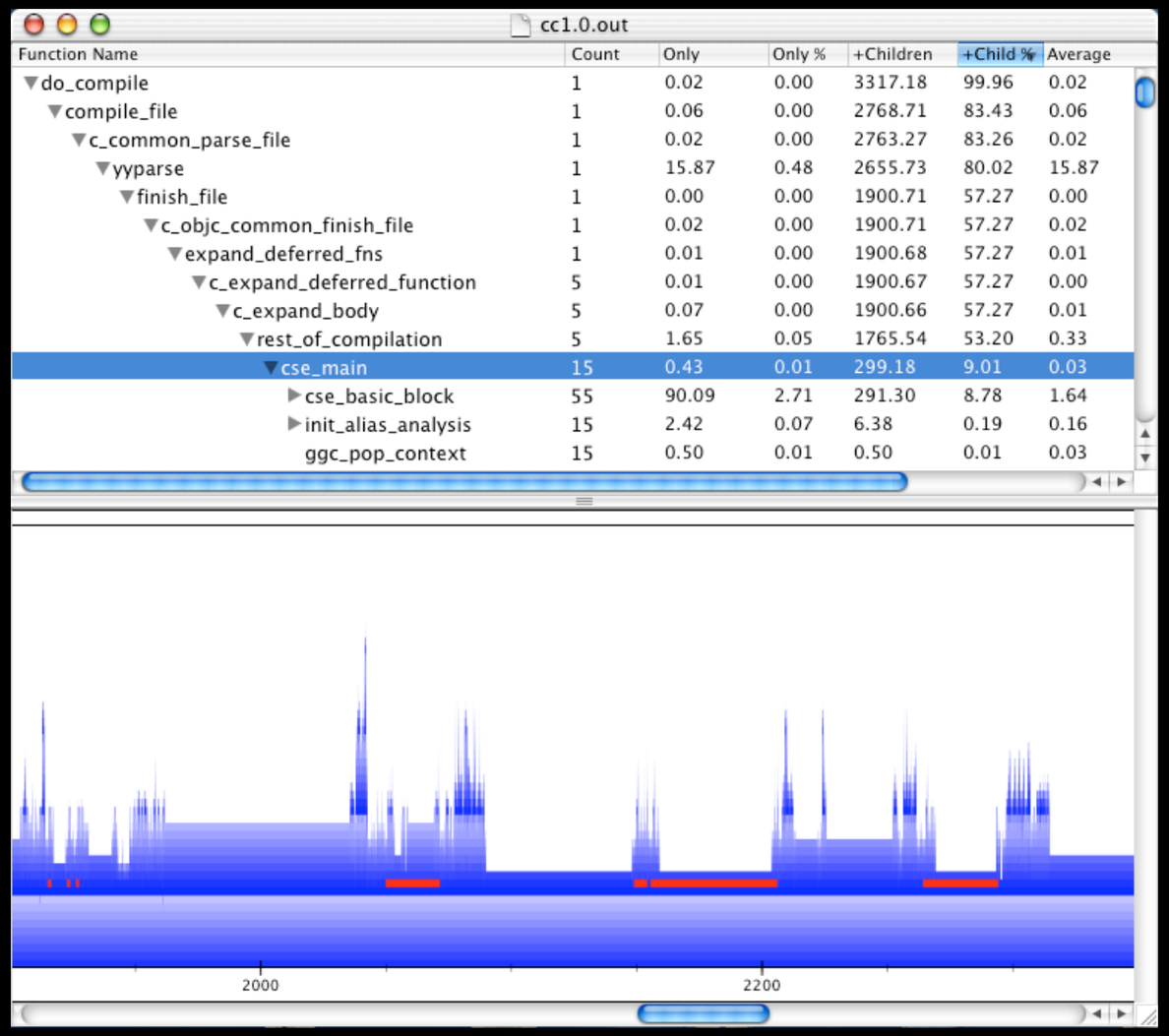
MONster

- Configure Performance Monitor Counters
- Collect PMC data based on
  - Timed intervals, event counts or hotkey
- View results in spreadsheet or chart form
- Shortcuts: Compute performance metrics
  - Memory bandwidth, cycles per instruction
  - Define your own with infix expressions
- Command line version
- Save and review sessions

Saturn



- Exact profile
- Visualize call tree
- Instruments functions at entry/exit with GCC
- Record function call history
- For each function:
  - Call counts
  - PMC event counts
  - Execution times



Wrap Up

Dave Payne

Future Performance Tools Directions

- Tight integration with Xcode
- Unification of various tools
- Changes to help focus on interesting data

Conclusion

- Tune your application to make best impression!
- Apple has tools to help
  - New: Spin Control, Activity Monitor
  - Improved: Sampler, Xcode integration, CHUD
  - More improvements to come!

How to Learn More

- Part of Xcode tools package
- Documentation for applications
  - Graphical tools: Documented in application
  - Command-line tools: Mac OS X man pages
  - /Developer/Documentation/Performance
- Inside Mac OS X books
  “Inside Mac OS X: System Overview”
- Apple Developer Connection tools pages
  http://developer.apple.com/tools/performance/

Who to Contact

Godfrey DiGiorgi
Technology Manager, Development Tools
ramarren@apple.com

Development Tools Engineering Feedback
xcode-feedback@group.apple.com

CHUD-Tools Engineering Feedback
chud-tools-feedback@group.apple.com

To Report Bugs
http://developer.apple.com/bugreporter

http://developer.apple.com/wwdc2003/urls.html

Roadmap

307 Xcode for Carbon Development
See how Carbon is supported by Xcode

Russian Hill
Thur., 9:00am

FF009 Dev Tools Feedback Forum
Give us your feedback

North Beach
Thur., 2:00pm

308 Debugging Services for Mac OS X
Hear the latest in debugging for Mac OS X

The Presidio
Thur., 3:30pm

310 Debugging and Tuning Carbon App
See specific Carbon debug and tune techniques

The Presidio
Fri., 9:00am

311 Software Testing Tools
Learn how to automate your testing practices

Russian Hill
Fri., 10:30am

Q&A

Godfrey DiGiorgi
Technology Manager, Development Tools
ramarren@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference